**Java Shortcuts:**

1.

Deque<int[]> stack = new ArrayDeque<int[]>(temperatures.length);

stack.push(new int[] { i, temperatures[i] });

2. List<Integer> integerlist = Arrays.asList(1,2);

4. list.toArray(new Object[0 or size]);

5.PriorityQueue<Map.Entry<Integer, Integer>> maxHeap =

new PriorityQueue<>((a,b)->(b.getValue()-a.getValue()));

6. Boundary check with Math.min

private int findNeighbors(int i, int j, int[][] board) {

int n = 0;

for (int x = Math.max(i - 1, 0); x <= Math.min(i + 1, board.length - 1) ; x++)

{

for (int y = Math.max(j - 1, 0); y <= Math.min(j + 1, board[0].length - 1); y++) {

if (!(i == x && j == y))

n = n + board[x][y];

}

}

return n;

}

7. for(int i=0;i<nums.length;i++)

{

if(i==0|| nums[i] != nums[i-1])

// print i

}

8.

HashMap<String, Set<String>> map = new HashMap<>();

Set<String> val1 = map.getOrDefault(w1, new HashSet<>());

val1.add(w2);

map.put(w1, val1);

9. Strings:

1. s.startsWith(str, i)
2. s.endsWith(str)
3. s.equals(str)
4. s.equalsIgnoreCase(str)
5. s.substring( i (inclusive), j (exclusive))
6. s.substring(I (inclusive) – untill end.
7. s.indexOf(str) -1, match not found. Else gives index.
8. s.lastIndexOf(str) – last match found.
9. S.trim()
10. S.toUpperCase()
11. S.toLowerCase()
12. S.charAt(i)
13. S.tocharArray()
14. S.replace(oldchar, newchar)
15. S.Split(“regex”)
16. S.split(“regex” , limit) – applied limit -1 times; so starts with 2

Java Collections:

1. TreeSet

**class** Dog **implements** Comparable<Dog> {

**int** size;

Dog(**int** s) {

size = s;

}

**public** **int** compareTo(Dog o) {

**return** size - o.size;

}

}

TreeSet<Dog> d = **new** TreeSet<Dog>();

d.add(**new** Dog(1));

d.add(**new** Dog(2));

d.add(**new** Dog(1));

TreeSet<Dog> d = **new** TreeSet<Dog>(**new** SizeComparator()); *// pass comparator*

d.add(**new** Dog(1));

d.add(**new** Dog(2));

d.add(**new** Dog(1))

1. HashSet

HashSet<Dog> dset = **new** HashSet<Dog>();

dset.add(**new** Dog(2));

dset.add(**new** Dog(1));

dset.add(**new** Dog(3));

dset.add(**new** Dog(5));

dset.add(**new** Dog(4));

Iterator<Dog> iterator = dset.iterator();

**while** (iterator.hasNext()) {

System.out.print(iterator.next() + " ");

}

1. LinkedHashSet

LinkedHashSet<Dog> dset = **new** LinkedHashSet<Dog>();

dset.add(**new** Dog(2));

dset.add(**new** Dog(1));

dset.add(**new** Dog(3));

dset.add(**new** Dog(5));

dset.add(**new** Dog(4));

Iterator<Dog> iterator = dset.iterator();

**while** (iterator.hasNext()) {

System.out.print(iterator.next() + " ");

}